Nom :
Prénom :
Groupe :

# I2 Devoir sur table

**Attention à bien indiquer vos nom, prénom, groupe sur chaque feuille**

*Aucun document autorisé*

## 1 Questions de cours (5 points)

**1) Expliquez brièvement à quoi correspondent les paramètres d'entrée, de sortie et d'entrée/sortie d'une procédure.**

voir le polycopié de cours.

**2) Écrivez en pseudo-code, par une structure, un type de donnée permettant de représenter dans une variable les informations suivantes sur un ouvrage : son titre, son nombre de pages, le nom de l'éditeur et celui des auteurs.**

**Type** Ouvrage = **Structure**
  nbPages : **Naturel**
  titre, editeur, auteurs : **Chaine de caracteres**
**finstructure**

**3) Écrivez en pseudo-code un type de donnée permettant de représenter par une seule variable une matrice ayant un nombre maximum de lignes et de colonnes déterminé par une constante *MAX* (ce nombre peut être inférieur à *MAX* et différent pour les lignes et les colonnes).**

**Type** Matrice = **Structure**
  valeurs : **Tableau**[1 .. 100][1 .. 50] **de Reel**
  taille1 : **Naturel**
  taille2 : **Naturel**
**finstructure**

**4) Qu'est-ce qu'une unité en Pascal ?**

voir le polycopié de cours.

**5) Expliquez brièvement à quoi correspondent les 3 modes d'ouverture de fichier que l'on peut réaliser en Pascal par les instructions *append*, *reset* et *rewrite*.**

voir le polycopié de cours.

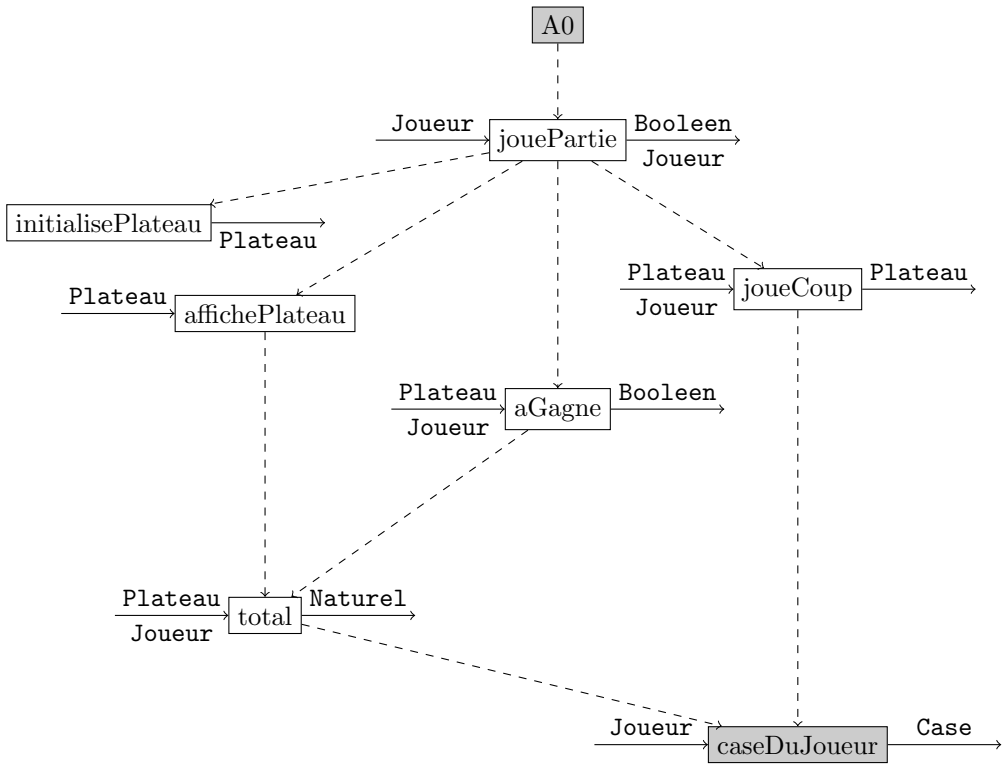
# 2 Écriture de code (9 points)

*Le jeu du quinze est un jeu de société se jouant à deux joueurs, notés traditionnellement X et O. Sur un tableau comportant les chiffres de un à neuf, chaque joueur à tour de rôle place un signe sur un chiffre qui n'a pas encore été joué. Le jeu comporte au maximum 3 tours de jeu. Pour gagner, il faut totaliser quinze en additionnant exactement trois chiffres joués par ce joueur.* (*Source Wikipedia*)

L'exemple suivant vous montre une fin de partie. Le joueur O a remporté la victoire car il a marqué 15 points en 3 prises, alors que le joueur X n'a que 13 points en 2 prises.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| O | | | | O | X | X | | O |

Nous allons réaliser une partie de la conception détaillée de ce programme en se basant sur l'analyse descendante suivante. Le type `Joueur` peut prendre l'une de ces deux valeurs : O ou X. Le type `Case` peut prendre l'une de ces trois valeurs : vide, prisParO, prisParX. Le type `Plateau` contient 9 valeurs de type `Case`.




Les sous-programmes correspondent aux signatures suivantes :

**procédure** jouePartie (**E** quiCommence : Joueur ; **S** ilYaGagnant : **Booleen** ; **S** leGagnant : Joueur)

Cette procédure englobe les actions de l'ensemble d'une partie. Elle prend en entrée quel joueur commence, indique en sortie s'il y a un gagnant au bout de 3 tours et quel joueur est gagnant.

**procédure** initialisePlateau (**S** unPlateau : Plateau)

Cette procédure remplit de vide un plateau.

**procédure** affichePlateau (**E** unPlateau : Plateau)

Cette procédure affiche les totaux de chaque joueur dans la partie en cours puis affiche le plateau.

**procédure** joueCoup (**E/S** unPlateau : Plateau ; **E** joueurEnCours : Joueur)

Cette procédure demande au joueur en cours quelle case il veut jouer et modifie le plateau en conséquence.

**fonction** aGagne (unPlateau : Plateau ; unJoueur : Joueur) **: Booleen**

Cette fonction calcule si le joueur a gagné, si c'est le cas elle renvoie `vrai` sinon `faux`.

**fonction** total (unPlateau : Plateau ; unJoueur : Joueur) **: Naturel**

Cette fonction calcule et retourne combien de points sont pris par le joueur.

**fonction** caseDuJoueur (unJoueur : Joueur) **:** Case

Cette fonction renvoie la valeur de `Case` correspondant au joueur. Par exemple, si on indique *O* en entrée alors la fonction renvoie *prisParO*.

# Questions

Définir en pseudo code les types `Joueur`, `Case` et `Plateau`.
**Type** Joueur = {joueurX, joueurO}
**Type** Case = {vide, priseX, priseO}
**Type** Plateau = **Tableau**[1..9] **de** Case
Écrire en pseudo code la conception détaillée des sous-programmes dont le fond est blanc sur l'analyse descendante.

**fonction** total (unPlateau : Plateau ; unJoueur : Joueur) **: Naturel**
  **Déclaration** somme,i : **Naturel** ; caseJoueur : Case
**debut**
  caseJoueur ← caseDuJoueur(unJoueur)
  somme ← 0
  **pour** i ←1 **à** 9 **faire**
    **si** unPlateau[i]=caseJoueur **alors**
      somme ← somme+i
    **finsi**
  **finpour**
  **retourner** somme
**fin**

**fonction** aGagne (unPlateau : Plateau ; unJoueur : Joueur) **: Booleen**
**debut**
  **retourner** total(unPlateau,unJoueur)=15
**fin**

**procédure** affichePlateau (**E** unPlateau : Plateau)
  **Déclaration** i : **Naturel** ; c : **Caractere**
**debut**
  **ecrire**("Total :")
  **ecrire**("Joueur O :", total(joueurO))
  **ecrire**("Joueur X :", total(joueurX))
  **ecrire**("")
  **pour** i ←1 **à** 9 **faire**
    **ecrire**(i," I ")
  **finpour**
  **pour** i ←1 **à** 9 **faire**
    **cas o** unPlateau[i] **vaut**
      *priseX*:
        c ← 'X'
      *priseO*:
        c ← 'O'
      *autre* :
        c ← ' '
    **fincas**
    **ecrire**(c," I ")
  **finpour**
**fin**

**procédure** joueCoup (**E/S** unPlateau : Plateau ; **E** joueurEnCours : Joueur)
  **Déclaration** i : **Naturel**
**debut**
  **repeter**
    **ecrire**(joueurEnCours," quelle case voulez-vous jouer ?")
    **lire**(i)

```
        si i=0 ou i>9  alors
            ecrire("La case doit être compris entre 1 et 9")
        finsi
        si unPlateau[i] !=vide  alors
            ecrire("La case doit être vide")
        finsi
    jusqu'a ce que i>0 et i<10 et unPlateau[i]=vide
    unPlateau[i] ← caseDuJoueur(joueurEnCours)
fin

procédure initialisePlateau (S unPlateau : Plateau)
    Déclaration  i : Naturel
debut
    pour i ←1 à 9 faire
        unPlateau[i] ← vide
    finpour
fin

procédure jouePartie (E quiCommence : Joueur ; S ilYaGagnant : Booleen ; S leGagnant : Joueur)
    Déclaration  i : Naturel ; joueurEnCours : Joueur ; lePlateau :Plateau
debut
    joueurEnCour ← quiCommence
    i ← 1
    ilYaGagnant ← faux
    initialisePlateau(lePlateau)
    affichePlateau(lePlateau)
    tant que i≤6 et ilYaGagnant=faux faire
        joueCoup(lePlateau,joueurEnCours)
        affichePlateau(lePlateau)
        ilYaGagnant ← aGagne(lePlateau,joueurEnCours)
        si ilYaGagnant alors
            leGagnant ← joueurEnCours
        finsi
        i ← i+1
        joueurEnCour ← succ(joueurEnCours)
    fintantque
fin
```

# 3 Analyse descendante (6 points)

Dans cet exercice, vous ne devez réaliser que la conception globale, c'est-à-dire écrire les **types de données** pertinents, une **analyse descendante** et **les signatures de fonctions et procédures avec un bref commentaire** pour expliquer leur rôle.

Le programme à concevoir est un programme de gestion de *playlist* musicale. Une playlist est un ensemble de pistes musicales (chansons) qui sont caractérisées par un titre, un nom d'artiste, et une durée (en secondes). Nous considérons dans cet exercice que nous ne gérons qu'une playlist, et que celle-ci est sauvegardée dans un fichier.

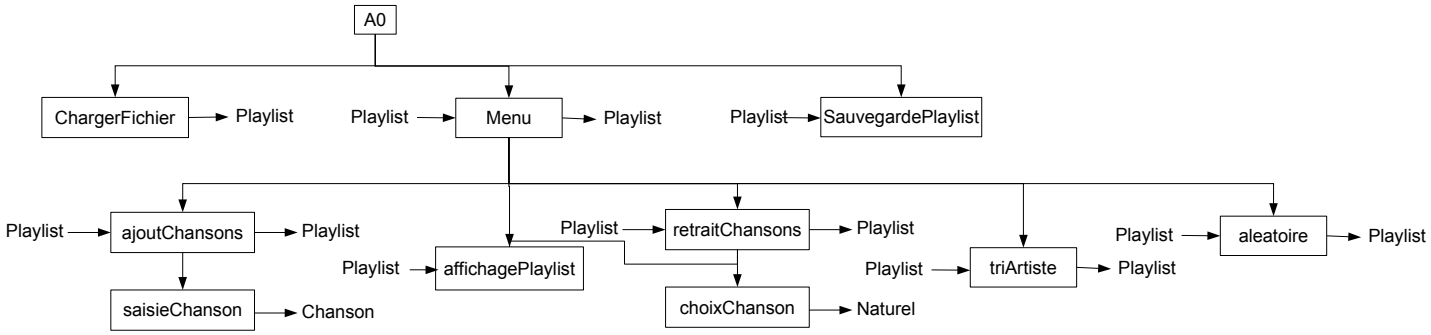
Au lancement du programme, la playlist en cours, stockée dans un fichier, est chargée. L'utilisateur peut alors choisir dans un menu d'ajouter une ou plusieurs chansons, d'afficher la playlist en cours, de supprimer une ou plusieurs chansons, de trier la playlist par artiste ou de la mettre en ordre aléatoire. L'ajout d'une chanson nécessite de saisir les différentes informations relatives à la chanson. La suppression d'une chanson implique de sélectionner une chanson parmi celles de la playlist. À la fin de l'exécution, la playlist modifiée est sauvegardée dans le même fichier qu'initialement.

**1) Définir deux types de données pour représenter une *chanson* et une *playlist*.**

**Type** Chanson = **Structure**
titre : Chaîne
auteur : Chaine
durée : Naturel
**finstructure**

**Type** Playlist = **Structure**
list : Tableau[1..MAX] de Chanson
nbChansons : Naturel
**finstructure**

**2) Dessiner une analyse descendante possible pour ce programme.**




**3) Donner la conception préliminaire (i.e. les signatures des sous-programmes) correspondant à cette analyse, en ajoutant un très bref commentaire indiquant le rôle de chaque sous-programme.**

**procédure** chargerFichier (**S** pl :Playlist){Cette procédure récupère la playlist en cours dans un fichier typé de Playlist fixe. Si le fichier n'existe pas, alors il est créé.}

**procédure** sauvegardeFichier (**E** pl :Playlist){Cette procédure enregistre la playlist en cours dans un fichier typé de Playlist fixe. }

**procédure** menu (**E/S** pl : Playlist){Cette procédure demande en boucle à l'utilisateur l'action qu'il souhaite réaliser, puis appelle la procédure dédiée.}

**procédure** affichagePlaylist (**E** pl :Playlist){Cette procédure affiche le contenu de la liste pl.}

**procédure** ajoutChansons (**E/S** pl : Playlist){Cette procédure permet d'ajouter des chansons à la Playlist en cours : tant que l'utilisateur souhaite entrer une chanson, la procédure saisieChanson est appelée et la chanson est ajoutée à la liste.}

**procédure** saisieChanson (**S** song : Chanson){Cette procédure demande à l'utilisateur les valeurs des champs d'une chanson.}

**procédure** retraitChansons (**E/S** pl : Playlist){Cette procédure permet de retirer des chansons de la Playlist en cours : tant que l'utilisateur souhaite retirer une chanson, la procédure affichagePlaylist est appelée pour donner à l'utilisateur la liste courante, puis choixChanson pour savoir l'indice à supprimer.}

**procédure** choixChanson (**S** num : Naturel){Cette procédure demande à l'utilisateur le numéro de la chanson à supprimer.}

**procédure** triArtiste (**E/S** pl : Playlist){Cette procédure effectue le tri des chansons en fonction du nom de l'artiste.}

**procédure** aleatoire (**E/S** pl : Playlist){Cette procédure repositionne de façon aléatoire les chansons dans la playlist}